



梦享考研系列

2016年考研 核心考点命题思路解密

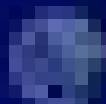
统考408核心题型

— Examination of 408 core questions —

梦享团队·编



北京邮电大学出版社
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS PRESS



清华大学出版社

2016年考研

核心考点命题思路精解

统考408核心题库

计算机组成原理、计算机操作系统、数据库系统原理、计算机网络

第1版 (2015)

清华大学出版社



梦享考研系列

2016 年考研核心考点命题思路解密

——统考 408 核心题型

梦享团队 编



北京邮电大学出版社
www.buptpress.com

内 容 简 介

《2016年考研核心考点命题思路解密——统考408核心题型》严格按照最新计算机考研408统考大纲编写,并精心将考研大纲细分成考点,有利于408统考和自主命题高校的考生抓住重点,着重训练。

本书每一个考点中的命题,绝大部分来源于历年名校计算机考研真题和408统考真题,少部分来源名校期末考试试题中的精华部分,是全国408统考大纲和高校考研真题的较好结合。为了提高考题的质量和解析的准确度,参考资料采用以考研权威教材、习题、考研真题为主,多方借鉴众多高校从事多年教育的教师课堂资料。梦享团队对每一个命题的思路和解题方法进行深入详细的讲解,并附上大量的图来帮助考生理解记忆,力求考生能够通过掌握一个题目而达到举一反三,有利于考生利用更少的时间掌握更多的知识。

本书可作为考生参加计算机专业研究生入学考试的备考复习用书,也可作为计算机专业的学生的习题集。

图书在版编目(CIP)数据

2016年考研核心考点命题思路解密:统考408核心题型/梦享团队编.--北京:北京邮电大学出版社,2015.8

ISBN 978-7-5635-4431-8

I. ①2… II. ①梦… III. ①电子计算机—研究生—入学考试—自学参考资料 IV. ①G643

中国版本图书馆CIP数据核字(2015)第176029号

书 名: 2016年考研核心考点命题思路解密——统考408核心题型

著作责任者: 梦享团队 编

责任编辑: 付兆华

出版发行: 北京邮电大学出版社

社 址: 北京市海淀区西土城路10号(邮编:100876)

发行部: 电话: 010-62282185 传真: 010-62283578

E-mail: publish@bupt.edu.cn

经 销: 各地新华书店

印 刷: 北京睿和名扬印刷有限公司

开 本: 787 mm×1 092 mm 1/16

印 张: 18.25

字 数: 473千字

版 次: 2015年8月第1版 2015年8月第1次印刷

ISBN 978-7-5635-4431-8

定 价: 39.80元

· 如有印装质量问题,请与北京邮电大学出版社发行部联系 ·

前 言

梦享团队队员以中科院、清华大学和北京交通大学三所高校的老师为主,其他名校老师为辅。另外还有在校的研究生,他们中有些人是考研论坛上参与答疑多年的版主等,目前团队33人。在考研复习和辅导上,梦享团队队员有着相对丰富的阅历。在考研的路上,梦享团队队员也经历过和大家一样的坎坷辛苦。我们深切地体会到,每一个考研的同学都十分不容易。

计算机考研的同学注重基础知识的掌握,更加看重实战能力。但目前的考研教材绝大多数倾向于知识点的讲解,不注重培养考生的实战能力,导致很多考生知识很丰富,但是很难将这些知识很好地运用于解题。编写偏向于实战的参考书不同于知识讲解,需要编者花费大量的时间来规划和布置章节、考点和解析考题。目前市面上现在能找到的计算机考研命题解析类参考资料,要么题目特别少但讲解特别详细啰嗦,要么题目太多的而对命题的讲解十分粗略甚至只有一个最终答案。因而,梦享团队决定写一套注重实战、解析详细、直击重点、严格遵照大纲的参考书。

经过两年多的努力,“梦享考研系列”参考书终于开始陆陆续续和大家见面了。回想过去两年多的日子,我们真正放松的日子很少,所有的时间都在没日没夜地学习和从事本套图书的编写。有时候一忙起来,整个团队都得熬夜到半夜三更。令人难以想象的是,两年多居然没有一个人说要退出,反而陆陆续续有人加入我们的团队。我们能够坚持,归根究底是因为,我们是编写图书的爱好者,期待向大家展现我们努力付出之后的成果。

为了提高图书的权威性,本套图书严格按照408统考大纲编写,涵盖了统考大纲所有指定的内容,并融合了历年名校考研真题的精华,是全国408统考大纲和高校考研真题的较好结合。为了提高考题的质量和解析的准确度,参考资料采用以考研权威教材、习题、考研真题为主,并多方借鉴众多高校从事多年教育的教师课堂资料。

本书具有以下特色。

1. 组织严谨,结构清晰

梦享考研系列图书通过对统考大纲和历年高校考研真题的深入剖析和总结,精心规划和部署了各个章节,对每一个章节的考点作了独家策划,使得本套图书组织严谨,结构清晰,便于大家对各章考点各个击破。

2. 突出重点,注重实战

对于每一个计算机专业的考研同学而言,时间非常有限。除了四门统考的繁重专业课之外,还有数学、英语和政治,复习工作量相当大。所以,突出重点,让同学们把极其有限的时间都花在刀刃上,是我们的首要工作;而提高同学们的实战能力,是我们系列图书的最终目的。

因此,在考题的挑选上,我们通过对统考和自主命题高校常出现的考题类型和知识点进行

深入总结,抛开在统考或者自主命题的考研真题上极少出现的极难、极易、极偏知识点,精心挑选了和考研难度相近的题型和考题供大家练习,以提高同学们的实战能力。

此外,我们还根据考点的重要程度来完成考点内容分布,在较重要的考点部署较多的内容,在较重要的内容部署较多的命题,在较为不重要的知识点抓住重点布置核心题型。目的也在于突出考试难度、突出考试重点,方便大家进行实战训练,提高学习效率,让考生在更短的时间内掌握更多的知识点。

3. 解析详细,深入剖析

“梦享考研”系列图书一共5本,每一本都很厚,可能会吓怕很多同学。是不是题目太多了?不是的,其实考题并不多,我们并不提倡题海战术,也不提倡对于同一个知识点反复命题和赘述,我们提倡“少而精”。针对每一个考点可能出现的命题类型,我们精心挑选了极具代表性的命题供大家实战训练,并对这些习题进行详细、深入的剖析,揭露问题的本质和解题的精髓,有助于大家掌握解题方法和技巧,提高大家的实战能力,在较短的时间内掌握更多的知识。

《2016年计算机考研核心考点命题思路解密》系列图书是我们团队汗水的结晶,融入了我团队的集体智慧。另外,真诚感谢我们团队新成员单开元、殷巧云、张丽方、胡明明、刘春、白洋等十几位同学提供的建议和帮助!

在接下来的时间里,我们会用我们最诚挚的心和最大的努力,给大家展示出更好的图书。我们每年都会合理调整这套图书,使得这套图书更加完善。

梦享团队会牢牢记住这样一句话——“助你们实现研究生梦想,是我们的梦想!”我们会伴随着2016年考研的同学一起度过艰辛的追梦季节,伴随着大家一起度过每一个艰辛的日日夜夜!也祝福2016年考研的你们,获得圆满的成功!

目 录

第一部分 数据结构部分

第 1 章 算法的时间复杂度和空间复杂度	2
第 2 章 线性表	6
2.1 线性表的定义和基本操作	6
2.2 线性表的实现	9
第 3 章 栈、队列和数组	16
3.1 栈和队列的基本概念	16
3.2 栈和队列的顺序存储结构	19
3.3 栈和队列的链式存储结构	22
3.4 栈和队列的应用	23
3.5 特殊矩阵的压缩存储	25
第 4 章 树与二叉树	28
4.1 树的概念	28
4.2 二叉树	29
4.3 树、森林	40
4.4 树的应用	42
第 5 章 图	45
5.1 图的概念	45
5.2 图的存储及基本操作	47
5.3 图的遍历	49
5.4 图的基本应用及其复杂度分析	50
第 6 章 查找	61
6.1 顺序查找法	61
6.2 折半查找法	61

6.3	B-树	65
6.4	散列(Hash)表及其查找	71
第7章	内部排序	76
7.1	插入排序	76
7.2	冒泡排序(bubble sort)	77
7.3	简单选择排序	78
7.4	希尔排序(shell sort)	79
7.5	快速排序	81
7.6	堆排序	83
7.7	归并排序(merge sort)	87
7.8	基数排序	88
7.9	各种内部排序算法的比较和应用	90

第二部分 计算机组成原理部分

第8章	计算机系统概述	94
8.1	计算机发展历程	94
8.2	计算机系统层次结构	95
8.3	计算机性能指标	95
第9章	数据的表示和运算	99
9.1	数值与编码	99
9.2	定点数的表示和运算	100
9.3	浮点数的表示和运算	104
9.4	算术逻辑单元 ALU	108
第10章	存储器层次机构	109
10.1	存储器的分类	109
10.2	存储器的层次化结构	110
10.3	半导体随机存取存储器	111
10.4	主存储器与 CPU 的连接	112
10.5	双口 RAM 和多模块存储器	113
10.6	高速缓冲存储器 Cache	116
10.7	虚拟存储器	118
第11章	指令系统	122
11.1	指令格式	122
11.2	指令的寻址方式	123

11.3 CISC 和 RISC 的基本概念	129
第 12 章 中央处理器(CPU)	131
12.1 CPU 的功能和基本结构	131
12.2 指令执行过程	132
12.3 数据通路的功能和工作原理	135
12.4 控制器的功能和工作原理	138
12.5 指令流水线	140
第 13 章 总线	144
13.1 总线概述	144
13.2 总线仲裁	147
13.3 总线操作和定时	149
13.4 总线标准	150
第 14 章 输入输出(I/O)系统	152
14.1 I/O 系统基本概念	152
14.2 外部设备	152
14.3 I/O 接口	154
14.4 I/O 方式	155

第三部分 操作系统部分

第 15 章 操作系统概述	162
15.1 操作系统的概念、特征、功能和提供的服务	162
15.2 操作系统的发展与分类	164
15.3 操作系统的运行环境	165
第 16 章 进程管理(进程与线程)	168
16.1 进程与线程	168
16.2 同步与互斥	171
第 17 章 进程管理(处理机调度与死锁)	184
17.1 处理机调度	184
17.2 死锁	188
第 18 章 内存管理	193
18.1 内存管理基础	193
18.2 虚拟内存管理	198

第 19 章 文件管理	206
19.1 文件系统基础	206
19.2 文件系统的实现	208
19.3 磁盘组织与管理	213
第 20 章 输入输出(I/O)管理	215
20.1 I/O 管理概述	215
20.2 I/O 核心子系统	217

第四部分 计算机网络部分

第 21 章 计算机网络体系结构	223
21.1 计算机网络概述	223
21.2 计算机网络体系结构与参考模型	223
第 22 章 物理层	228
22.1 通信基础	228
22.2 传输介质	231
22.3 物理层设备	231
第 23 章 数据链路层	233
23.1 数据链路层的功能	233
23.2 组帧	234
23.3 差错控制	235
23.4 流量控制与可靠传输机制	237
23.5 介质访问控制	238
23.6 局域网	242
23.7 广域网	243
23.8 数据链路层设备	244
第 24 章 网络层	247
24.1 网络层的功能	247
24.2 路由算法	249
24.3 IPv4	249
24.4 IPv6	256
24.5 路由协议	257
24.6 IP 组播和移动 IP	258
24.7 网络层设备	258

第 25 章 传输层	260
25.1 传输层提供的服务	260
25.2 UDP 协议	261
25.3 TCP 协议	263
第 26 章 应用层	269
26.1 网络应用模型	269
26.2 DNS 系统	270
26.3 FTP	272
26.4 电子邮件	273
26.5 WWW	275
参考文献	278



第一部分

数据结构部分

第 1 章

算法的时间复杂度和空间复杂度



温馨提示:

算法的时间复杂度和空间复杂度几乎在每一年的统考中都会命中一个题,基本的要求是考生能够根据算法来分析时间复杂度。

1. 设 n 是描述问题规模的非负整数,下面程序片段的时间复杂度是()。

```
x = 2;
```

```
while(x < n/2)
```

```
    x = 2 * x;
```

- A. $O(\log_2 n)$ B. $O(n)$ C. $O(n \log_2 n)$ D. $O(n^2)$

【2011 年统考——第 1 题】

【考查内容】算法的时间复杂度。

【解析】首先,我们要找到程序中执行频率最高的语句,来计算算法的时间复杂度。显然,语句“ $x=2 * x$ ”的执行频率最高。假设该语句总共执行了 k 次,那么第 $k+1$ 次执行时因为 $2^{k+1} \geq n/2$ 而退出执行。

值得注意的是,在计算算法的时间复杂度时,通常我们都会忽略系数和抓大端。比如某个算法的执行次数与 n 的关系为 $50n^2 + 100n + 10\ 000$,抓大端时我们只考虑阶最高的项,即 $50n^2$,然后再删除系数 50,得到算法的时间复杂度为 $O(n^2)$ 。

这里一样的道理,有 $k+1 \approx \log_2(n/2)$,程序的执行次数 k 约为 $\log_2(n/2) - 1$,算法的时间复杂度为 $O(\log_2 n)$ 。

【参考答案】A

2. 求整数 $n(n \geq 0)$ 阶乘的算法如下,其时间复杂度是()。

```
int fact(int n)
```

```
{
```

```
    if (n <= 1) return 1;
```

```
    return n * fact(n - 1);
```

```
}
```

- A. $O(\log_2 n)$ B. $O(n)$ C. $O(n \log_2 n)$ D. $O(n^2)$

【2012 年统考——第 1 题】

【考查内容】算法的时间复杂度。

【解析】时间复杂度是由语句频度分析得来。递归算法中重复执行的语句主要是调用，所以递归算法的时间复杂度分析主要是分析递归算法中递归函数调用的次数。

对于本题，咱们给出一个递归的图解过程，如图 1.1 所示。

从图解可知，该递归过程是线性的。总共递归执行了 n 次，时间复杂度为 $O(n)$ 。

【参考答案】B

3. 已知两个长度分别为 m 和 n 的升序链表，若将它们合并为一个长度为 $m+n$ 的降序链表，则最坏情况下的时间复杂度是()。

- A. $O(n)$ B. $O(m \times n)$ C. $O(\min(m, n))$ D. $O(\max(m, n))$

【2013 年统考——第 1 题】

【考查内容】两个升序链表合并成降序链表的时间复杂度。

【解析】我们假设两个升序链表 A 和 B 分别如图 1.2 所示。图中会出现时间复杂度最大的情况(请同学自己想想还有没有其他情况?)，即 B 表全部插入之后，A 表剩余的结点采用头插法插入到链表 C 中。

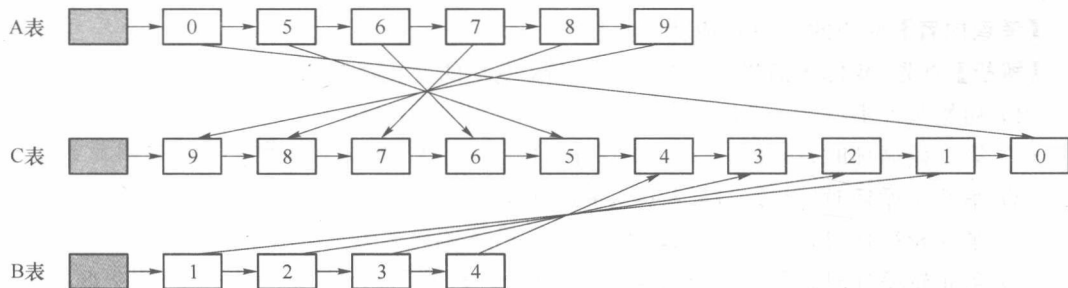


图 1.2

要将图 1.2 的 A 表和 B 表合并成 C 表，显然时间复杂度是 $O(m+n)$ 。但是没有这个选项，所以需要咱们再考虑一下哪一个更加接近这个答案。我们再假设一种极端的情况，那就是图 1.2 的一个延伸。假设 A 表的长度远比 B 表长(即 m 远大于 n)，那么时间复杂度 $O(m+n)$ 等价于 $O(m)$ 。故而，将长度为 m 和 n 的升序链表合并成长度为 $m+n$ 的降序链表，时间复杂度为 $O(\max(m, n))$ 。

实际上， $\max(m, n) < m+n < 2\max(m, n)$ ，有 $O(m+n)$ 等价于 $O(\max(m, n))$ 。这样可能便于大家理解一些。

【参考答案】D

4. 下列程序段的时间复杂度是()。

```
count = 0;
for(k = 1; k <= n; k * = 2)
    for(j = 1; j <= n; j++)
        count++;
```

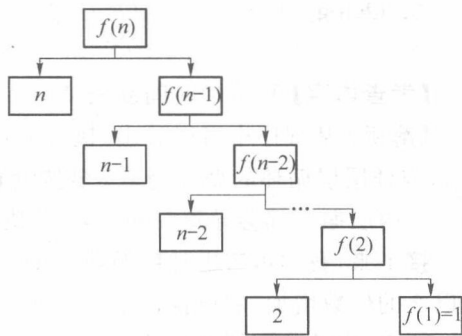


图 1.1

A. $O(\log_2 n)$

B. $O(n)$

C. $O(n \log_2 n)$

D. $O(n^2)$

【2014年统考——第1题】

【考查内容】算法的时间复杂度。

【解析】从程序中可以看出,执行频度最高的语句是“count++”。内层循环的循环变量是 j ,与外层层循环的循环变量 i 显然没有关系。从内层循环“for($j=1; j \leq n; j++$)”来看,每一个内层循环都会导致“count++”执行 n 次。

接下来,我们再考虑外层循环“for($k=1; k \leq n; k *= 2$)”。每一次执行外层循环, k 都是以 2 的倍数增加,假设执行了 t 次,则 $2^t = n$,显然有 $t = \log_2 n$ 。

那么,内外两层循环总共执行了 $n * t$ 次,即 $n \log_2 n$ 次。

【参考答案】C

5. 下面程序段的时间复杂度是()。

```
i = s = 0;
while(s < n)
{
    i++;
    s += i;
}
```

A. $O(\log n)$

B. $O(\sqrt{n})$

C. $O(n)$

D. $O(n^2)$

【2010年——湖南大学】

【考查内容】算法的时间复杂度。

【解析】首先,我们来简单地分析一下 i 和 s 变量的变化情况:

(1) 初始条件下, $i = s = 0$;

(2) 第 1 轮循环时, $i = 1, s = s + i = 0 + 1 = 1$;

(3) 第 2 轮循环时, $i = 1 + 1 = 2, s = s + i = 1 + 2 = 3$;

(4) 第 3 轮循环时, $i = 2 + 1 = 3, s = s + i = 3 + 3 = 6$;

(5) 第 4 轮循环时, $i = 3 + 1 = 4, s = s + i = 6 + 4 = 10$;

(6) ...

(7) 第 t 轮循环是, $i = t, s = 1 + 2 + 3 + \dots + t = t(t+1)/2$ 。

假设在 $t+1$ 轮时退出循环,根据判断条件 $s < n$ 可知, $t(t+1)/2$ 需小于等于 n 。根据抓大端的规则, $t^2 < n$ 必须成立。所以,算法的时间复杂度为 $O(\sqrt{n})$ 。

【参考答案】B

6. 下面算法的空间复杂度为()。

```
float aver(float a[], int n)
{
    int j;
    for(j = n; j > 0; j--)
        printf("%8.2f", a[j]);
}
```

A. $O(1)$

B. $O(\log_2 n)$

C. $O(n)$

D. $O(n^2)$

【2013年——昆明理工大学】

【考查内容】算法的时间复杂度。

【解析】本题是我们求算法的时间复杂度一类题目中比较简单的题目了。本题只涉及单层循环。算法开始执行时, $j=n$, 此后每执行一次循环, j 减 1, 当 $j=0$ 时退出。显然, 循环执行了 n 次。所以, 时间复杂度为 $O(n)$ 。

【参考答案】 C

7. 若一个算法的时间复杂度用 $O(n)$ 表示, 其中 n 的含义是()。

- A. 循环层数 B. 语句条数 C. 问题规模 D. 函数数量

【2012 年——青岛理工大学】

【考查内容】时间复杂度 $O(n)$ 中 n 的含义。

【解析】时间复杂度 $O(n)$ 中, n 称为问题的规模, 当 n 不断变化时, 时间频度 $T(n)$ 也会不断变化。一般情况下, 算法中基本操作重复执行的次数是问题规模 n 的某个函数, 用 $T(n)$ 表示。

【参考答案】 C

8. 下列程序段的时间复杂度为()。

```
for(i = n - 1; i >= 1; i--)  
    for(j = 1; j <= i; j++)  
        if(DATA[j] > DATA[j + 1]) {  
            TEMP = DATA[j];  
            DATA[j] = DATA[j + 1];  
            DATA[j + 1] = TEMP;  
        }
```

- A. $O(n)$ B. $O(n^2)$ C. $O(n^3)$ D. $O(n \log_2 n)$

【2013 年——江苏大学】

【考查内容】算法的时间复杂度。

【解析】本题是一个很简单的冒泡算法。每一次冒泡算法执行结束, 都有一个当前未有序的序列中最大的元素落在未有序序列的末尾位置, 有序的表长度加 1。

我们可以看出, if 语句的时间复杂度是由两层循环决定的。外层循环执行 $n-1$ 次; 内层循环的次数与 i 有关。具体的执行次数如下:

- (1) 当 $i=n-1$ 时, $j=1, 2, 3, \dots, i$ 总共执行 $n-1$ 次;
- (2) 当 $i=n-2$ 时, $j=1, 2, 3, \dots, i$ 总共执行 $n-2$ 次;
- (3) ...

显然, 程序总共执行了 $1+2+3+4+\dots+(n-2)+(n-1)=n(n-1)$ 次, 其时间复杂度为 $O(n^2)$ 。

【参考答案】 B

第 2 章

线性表

2.1 线性表的定义和基本操作



温馨提示:

线性表的定义和基本操作,从历年统考和高校自主命题的真题来看,主要考察了三个问题:1.什么是线性表? 2.线性表存储方式? 3.线性表怎么样插入、删除? 请同学们围绕这三个问题,展开复习,其中第3个问题是本考点的重点。

1. 在一个单链表中,已知 *q 结点是 *p 结点的前驱结点,若在 *q 和 *p 之间插入 *s 结点,则执行()。

- A. $s \rightarrow next = p \rightarrow next; p \rightarrow next = s;$
- B. $p \rightarrow next = s \rightarrow next; s \rightarrow next = p;$
- C. $q \rightarrow next = s; s \rightarrow next = p;$
- D. $p \rightarrow next = s; s \rightarrow next = q;$

【2013 年——昆明理工大学】

【考查内容】在单链表中插入结点的正确操作。

【解析】单链表中插入和删除结点是常见的操作,希望大家要多练习,掌握这些基本的操作。说透了,其实不同的题目中,这些操作都大同小异,学会灵活应用,才是硬道理。

已知 *q 结点是 *p 结点的前驱结点,在 *q 结点和 *p 结点之间插入 *s 结点,如图 2.1 所示。

注意插入结点的过程中,要防止“掉链”的情况(即指针调整的过程中,后续指针出现错误)。为了防止“掉链”,需要把 q 的指针先赋给 s 的指针,然后再调整 q 的指针指向结点 *s。即“ $s \rightarrow next = q \rightarrow next; q \rightarrow next = s;$ ”。

当然,若 3 个结点的指针都很明确,像本题三个结点的指针 p、q、s 均已知的情况下,可以

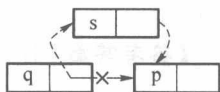


图 2.1